Université de Montréal

Insertion tardive du contenu sémantique des traits fonctionnels

par Emmanuel Parenteau

Département de linguistique et de traduction
Faculté des arts et des sciences

Mémoire présenté
en vue de l'obtention du grade de Maîtrise ès arts (M.A.)
en linguistique

Août 2018

Résumé

Je propose que le contenu sémantique des catégories fonctionnelles soit inséré post-syntaxiquement et cycliquement, de façon similaire à ce qui est proposé pour le contenu phonologique en nanosyntaxe. Dans cette approche, le lexique pré-syntaxique est composé exclusivement d'indices dénués de contenu phonologique ou sémantique. L'opération Merge est libre, et les constituants syntaxiques sont directement lexicalisés (*Phrasal Spell-Out*) de façon indépendante à chacune des interfaces. Comme alternative à une Séquence Fonctionnelle (Fseq) en tant que contrainte syntaxique sur les outputs de Merge (Starke 2001), je propose une façon de réencoder la Fseq en termes de sélection sémantique. Je montre ensuite comment l'Universel 20 (Greenberg 1963; Cinque 2005) peut être dérivé sans stipuler la Fseq en syntaxe. Cette architecture simplifie l'interface à PF et permet une analyse élégante de divers phénomènes morphologiques non concaténatifs tels que l'apophonie et l'infixation en éliminant le besoin pour des règles de réajustement.

Mots-clés : Conditions d'interfaces, Nanosyntaxe, Insertion tardive, Lexicalisation des constituants, Universel 20 de Greenberg

Abstract

I propose that the semantic content of functional categories be inserted post-syntactically and cyclically, in a manner analogous to what is assumed within nanosyntax for the phonological content. Under this approach, the pre-syntactic lexicon is comprised exclusively of indices devoid of phonological or semantic content. The Merge operation applies freely, and the syntactic constituents are directly spelled out (*Phrasal Spell-Out*) in an independent manner at each interface. As an alternative to a Functional Sequence (Fseq) *qua* syntactic constraint on the outputs of Merge (Starke 2001), I propose a way to reencode the Fseq in terms of semantic selection. I then show how Universal 20 (Greenberg 1963; Cinque 2005) can be derived without stipulating the Fseq in syntax. This architecture simplifies the interface at PF and provides an elegant analysis for various non concatenative morphological phenomena such as apophony and infixation, obviating the need for readjustment rules.

Keywords : Interface conditions, Nanosyntax, Late Insertion, Phrasal Spell-Out, Greenberg Universal 20

Table des matières

Liste des sigles et abréviations

DM : Morphologie distribuée

EP : *Elsewhere Principle*

Fh : Hiérarchie fonctionnelle

Fseq : Séquence fonctionnelle

IC : Idéal concaténatif

LCA : Axiome de correspondance linéaire

NS : Nanosyntaxe

OFOH : *One feature One Head*

PSO : Lexicalisation des constituants (*Phrasal Spell-Out*)

SMT : Thèse minimaliste forte (*Strong Minimalist Thesis*)

SP : *Superset Principle*

Remerciements

Je tiens à remercier ma directrice Christine Tellier pour son aide et ses conseils à travers déjà plusieurs années, pour m'avoir sorti du pétrin à maintes reprises, et pour avoir toujours cru en mes idées.

Un grand merci à Tom Leu pour son stimulant séminaire de nanosyntaxe et pour ses précieux conseils et commentaires qui ont constitué un apport majeur à ce travail.

Merci également à mes vieux amis; Carolyne pour son soutien et ses encouragements dans les moments difficiles; Thierry et Noé pour des années de conversations stimulantes qui ont contribué à former ma pensée et m'ont donné envie de faire des études supérieures.

Enfin et surtout, merci à Marie et Robert, mes parents, pour leur amour inconditionnel et leur appui continuel sans lesquels je n'aurais jamais pu venir à bout de ce mémoire, dont ils sont, j'en suis sûr, beaucoup plus fiers que je ne le suis moi-même.

Introduction

Dans un esprit radicalement minimaliste, ce travail cherche à déterminer la nature des objets élémentaires à la base de la dérivation, les traits syntaxiques, en prenant le plus littéralement possible l'idée que le contenu de la syntaxe se limite à ce qui est absolument nécessaire pour coordonner les représentations sémantique et phonologique. Ma principale proposition est que le contenu qu'il faut attribuer aux traits pour remplir cette fonction est beaucoup plus minimal qu'on le suppose habituellement. L'architecture que je propose va aussi à l'encontre de l'idée largement reçue dans la littérature minimaliste que la Faculté du langage entretient une relation plus étroite avec C-I qu'avec S-M, et que la syntaxe satisfait optimalement les contraintes de la sémantique au prix de certaines imperfections à l'interface phonologique. Dans le chapitre 1, je présente les présupposés et les outils théoriques qui formeront la base de mon analyse. Le chapitre 2 défend l'hypothèse que le contenu sémantique est inséré post-syntaxiquement, en montrant qu'une telle division permet une analyse simplifiée de certains processus morphologiques comme l'apophonie. Le chapitre 3 montre comment la production de Merge peut être contrainte indirectement en exploitant les conditions des interfaces. Le chapitre 4 formule plus explicitement les bases théoriques de mon approche et utilise les outils ainsi proposés pour analyser l'infixation et le ablaut de façon similaire à l'apophonie au chapitre 2.

Chapitre 1 Considérations théoriques

1.1 Merge et la Thèse minimaliste forte

Le programme minimaliste a comme objectif de déterminer dans quelle mesure les propriétés de la faculté du langage peuvent être expliquées par, ou réduites à, des facteurs externes: les contraintes de lisibilité imposées par les interfaces avec les composantes sémantique (C-I) et phonologique (S-M), et les considérations d'économie computationnelle. L'hypothèse par défaut est donc que la grammaire inclut ni plus ni moins ce qui est absolument nécessaire pour satisfaire les conditions d'interfaces.

(1) Thèse minimaliste forte / *Strongest Minimalist Thesis* (SMT)

> *Language is an optimal solution to legibility conditions.*
> (Chomsky 2000:96)

> *The principles of language are determined by efficient computation and language keeps to the simplest recursive operation, Merge, designed to satisfy interface conditions in accord with independent principles of efficient computation.*
> (Berwick & Chomsky 2011: 30)

La grammaire doit comporter minimalement une opération combinant récursivement des objets syntaxiques (Merge). La formulation de Merge varie en complexité dans la littérature, par exemple selon qu'elle inclut ou non un mécanisme de projection fournissant une étiquette à son output, ou selon que son application est libre ou limitée à des instances de vérification de traits. Comme l'objectif est ici de réduire au minimum le contenu de la syntaxe, j'adopterai la

position de Collins (2017) en prenant la version la plus simple possible de Merge, soit une opération libre et dont l'output ne reçoit pas d'étiquette.

(2) Merge (A,B) = {A,B} où A et B sont des objets syntaxiques

Le domaine d'application de Merge est l'ensemble des objets syntaxiques, qui se définit récursivement comme l'union du lexique pré-syntaxique (LEX) et de l'ensemble des outputs de Merge. Je soutiens que cette version de Merge est suffisante pour alimenter correctement les interfaces et que celles-ci sont en mesure de filtrer la surgénération massive de Merge.

Comme Merge, le contenu de LEX devrait également être déterminé par rapport à la SMT, c'est-à-dire que la nature des objets à la source des dérivations syntaxiques devrait être définie de façon explicite et motivée en référence à l'objectif d'alimenter optimalement les interfaces. D'abord, pour éviter toute redondance, Merge devrait être considérée comme la seule opération génératrice de structure. La conception des éléments lexicaux en tant que baluchons de traits (*feature bundles*), adoptée par exemple en théorie des phases (Chomsky 2000, 2001, 2013, 2015) devrait par conséquent être évitée, puisque ces baluchons sont des structures de traits et doivent donc être construits par Merge. Autrement dit, les baluchons de traits seront considérés ici comme des structures formées dans la syntaxe plutôt que comme des objets au statut distinct et assemblés pré-syntaxiquement (voir Boeckx 2015 pour une argumentation similaire contre un assemblage pré-syntaxique des traits). Une première conclusion à laquelle nous mène la SMT en ce qui concerne LEX est donc que ses éléments devraient être atomiques (dépourvus de structure interne). L'atomicité des éléments lexicaux est une caractéristique centrale de la conception communément appelée OFOH (One feature one

head), adoptée notamment en Cartographie et en Nanosyntaxe (Starke 2009), selon laquelle chaque trait morphosyntaxique a sa propre projection. Le contenu de LEX serait donc égal à l'ensemble des traits morphosyntaxiques. Chacun de ces traits correspondrait par ailleurs à un contenu sémantique de classe fermée (catégorie fonctionnelle). On peut alors parler de traits syntactico-sémantiques puisque les mêmes éléments sont visibles pour la syntaxe et la sémantique. Une conséquence est que la relation entre ces deux modules est beaucoup plus directe que celle entre la syntaxe et S-M; l'interface à C-I consiste à *transférer* les mêmes trait d'un module à l'autre, alors que l'interface à S-M nécessite de convertir, via des entrées lexicales, les structures syntaxiques en représentations phonologiques. Conceptuellement, le recoupement des traits élémentaires entre représentations syntaxiques et sémantiques apparaît toutefois suspect, en particulier du point de vue de la modularité. Par définition, un trait est un élément interprétable et manipulable par un certain ensemble de règles ou d'opérations (i.e. un certain module). Affirmer une équivalence entre traits syntaxiques et sémantiques revient donc à traiter la syntaxe et C-I comme un seul et même module, ce qui contrevient à l'idée que la syntaxe se limite à Merge. Au cours de ce travail, j'apporterai d'autres arguments contre la stipulation d'une relation 1:1 entre traits syntaxiques et sémantiques en montrant qu'elle va au-delà de ce qui est absolument nécessaire pour alimenter correctement les interfaces, et que son abandon permet de simplifier l'interface à S-M.

1.2 Hiérarchie fonctionnelle et séquence fonctionnelle

Pour rendre compte des contraintes observées à travers les langues dans la distribution des traits syntactico-sémantiques (Hiérarchie Fonctionnelle (Cinque 1999; Rizzi 1997)), les cartographes et les nanosyntacticiens ont recours au concept de Séquence Fonctionnelle (Fseq) (voir entre autres Starke 2001, Cinque 2005), c'est-à-dire un ordre rigidement stipulé dans la G.U. suivant lequel les traits entrent dans la dérivation. Le principe en (3) rend compte de la Hiérarchie Fonctionnelle en stipulant l'ordre dans lequel les traits sont combinés par Merge.

(3) a. Fseq = $<F_1, F_2, \ldots, F_n>$
b. F_n doit entrer dans la dérivation après F_{n-1} (pour tout n>1)

L'ordre de base permis par la Fseq sert ensuite d'input à des opérations de mouvement hautement contraintes censées générer tous et seulement les ordres attestés à travers les langues (voir p. ex. Cinque 2005). Pour la discussion qui suit, il sera crucial de faire la distinction entre d'une part la Hiérarchie Fonctionnelle en tant que généralisation empirique observée dans la distribution des traits fonctionnels à travers les langues, et d'autre part le principe en (3), qui est un moyen parmi d'autres concevables de dériver la Hiérarchie Fonctionnelle. Je ferai désormais référence à la généralisation empirique par le terme de *Hiérarchie Fonctionnelle* (Fh), et au principe en (3) par celui de *Séquence Fonctionnelle* (Fseq). La Fseq est une contrainte purement syntaxique qui ferait en principe partie intégrante de la G.U., à côté de Merge. Toutefois, la richesse manifestée par le nombre et la spécificité des traits fonctionnels rend peu plausible que la Fseq fasse partie d'une faculté spécifiquement

langagière, tant d'un point de vue neurologique qu'évolutif. Plusieurs auteurs ont en fait proposé que la Fh est au moins en partie dérivable de facteurs sémantiques (Ernst 2002, Nilsen 2003, Ramchand et Svenonius 2014). La perspective de reléguer à la sémantique le fardeau porté par la Fseq est intéressante non seulement parce qu'elle réduit automatiquement le contenu de la G.U., mais aussi et surtout parce qu'elle permet de radicalement réévaluer le contenu de LEX par rapport à la SMT. Si l'application de Merge est libre et que la seule fonction de la syntaxe est d'assurer la coordination entre les représentations sémantiques et phonologiques, le recoupement entre traits syntaxiques et sémantiques apparaît étonnant et devrait être justifié empiriquement.

1.3 L'interface phonologique

Je prendrai comme point de départ les outils utilisés en NS et apporterai certaines modifications au cours du travail (voir Baunaz & Lander 2018, Starke 2009 pour des introductions). L'interface avec S-M a essentiellement deux fonctions :

i) la lexicalisation s'occupe d'identifier et de convertir certains morceaux de la structure syntaxique en représentations phonologiques via les entrées lexicales spécifiques à la langue. L'insertion lexicale en NS peut se concevoir comme l'application d'une règle de réécriture non contextuelle. Elle se distingue donc de l'insertion en Morphologie Distribuée (DM) où la forme insérée peut être sensible au contexte syntaxique.

(4) Insertion Lexicale

a. En NS: $S \rightarrow P$

b. En DM: $S_i \rightarrow P / (S_j) \_ (S_k)$

où S est un objet syntaxique et P une représentation phonologique

ii) la linéarisation détermine les relations de précédence entre les représentations obtenues par insertion selon leur configuration syntaxique via l'Axiome de Correspondance Linéaire (LCA, Kayne 1994)[1]. L'insertion et le LCA constituent des contraintes imposées par l'interface phonologique sur l'output de la syntaxe: celle-ci doit produire des objets différenciables par les entrées lexicales de la langue de sorte que chaque entrée peut s'appliquer de façon non ambiguë. Elle doit aussi établir entre ces objets des asymétries structurales que le LCA peut interpréter comme asymétries linéaires (i.e. comme relations de précédence), de façon non ambiguë.

La lexicalisation doit en outre rendre compte de la flexibilité de la relation entre les traits de la structure syntaxique et leurs réalisations phonologiques (i.e. syncrétismes et formes supplétives). Le spell-out en NS se distingue par la Lexicalisation des Constituants (*Phrasal Spell-Out*, ci-après PSO), qui permet l'insertion lexicale dans des nœuds non terminaux (une autre différence fondamentale avec DM), puis par le principe connexe de Superset Principle (SP), qui détermine dans quelles conditions une entrée lexicale est admissible pour réaliser un

[1] Informellement, le LCA convertit une asymétrie structurale en asymétrie linéaire de sorte que A précède B si et seulement si A c-commande B.

constituant syntaxique donné. L'attrait du PSO est autant conceptuel qu'empirique. Conceptuellement, le PSO retire la stipulation tacite que le spell-out est limité aux nœuds terminaux, et simplifie l'interface à S-M en éliminant le besoin de postuler et coordonner une série d'allomorphes (incluant des allomorphes zéro) pour rendre compte des cas de syncrétisme et de supplétion (voir Starke 2011a). Le PSO est également plus restrictif et fait de meilleures prédictions en ce qui concerne certains patrons de syncrétismes non attestés à travers les langues, appelés *ABA (voir par exemple Caha 2013 sur les syncrétismes dans les paradigmes casuels). Je supposerai dans ce qui suit que l'insertion lexicale peut s'effectuer sur tout nœud syntaxique, terminal ou non.

En somme, l'insertion lexicale consiste à associer une structure syntaxique à une réalisation phonologique via des entrées lexicales dont la forme est illustrée en (5), où /p/ représente une forme phonologique et {F{G{H}}} représente une structure de traits syntaxiques de taille quelconque. Par convention, on appelle {F{G{H}}} en (5) le L-Tree (arbre lexical) de l'entrée lexicale, par opposition au S-Tree, qui est la structure de traits formée dans la syntaxe et devant être réalisée par une entrée lexicale.

(5) </p/, {F{G{H}}}>

Suivant le Superset Principle (SP), donné en (6), une entrée lexicale est admissible pour réaliser non seulement un S-Tree identique à son L-Tree, mais aussi tout sous-constituant de celui-ci.

(6) Superset Principle
A lexical tree L can match a syntactic tree S if L is a superset (proper or not) of S. L matches S if L contains a node that is identical to a node in S and all the nodes below are also identical.

Par exemple, le SP permet à l'entrée (5) de lexicaliser les S-Trees {F{G{H}}}, {G{H}} ou {H} (mais pas {F}, {G}, {F{G}} ou {F{H}}}). Enfin, le Elsewhere Principle (EP). donné en (7), résout les cas où il y a compétition entre plusieurs L-Trees.

(7) Elsewhere Principle
If more than one L-tree can lexicalize the same S-tree (by the Superset Principle), then the L-tree with the least amount of superfluous material is chosen.

En somme, le SP et le EP assurent qu'une structure syntaxique sera réalisée par l'entrée lexicale dont le L-Tree forme son plus petit surensemble. Pour voir comment ces mécanismes permettent d'analyser les syncrétismes, prenons l'exemple des interprétations allative et locative de la préposition *à* en français, correspondant respectivement à *to* et *at* en anglais.

(8)

Aller à la plage	(Allatif)
Demeurer à la plage	(Locatif)
Go to the beach	(Allatif)
Stay at the beach	(Locatif)

Le français montre donc un syncrétisme entre le locatif et l'allatif, contrairement à l'anglais. D'après Pantcheva (2011), la structure de l'allatif serait plus complexe que celle du locatif, et inclurait proprement cette dernière.

(9) S-Trees pour Locatif et Allatif en français et en anglais

a. [Path[Place]]	Allatif
b. [Place]	Locatif

La différence entre l'anglais et le français en ce qui concerne ces prépositions peut alors s'expliquer par les entrées lexicales en (10)[2].

(10) Entrées lexicales

a. </à/, [Path[Place]] > b. </to/, [Path[Place]]>

c. </at/, [Place]>

En français, le SP permet à l'entrée (10a) de réaliser les deux S-Trees en (9a-b), et l'absence d'une entrée plus spécifique pour (9b) donne un syncrétisme. L'anglais comporte en revanche une entrée lexicale correspondant à chacune des deux structures en (9). Le SP permet aux deux entrées en (10b-c) de lexicaliser la structure du locatif en (9b), mais le EP résout la compétition en privilégiant le plus petit surensemble, soit (10c).

La nécessité de lexicaliser la structure syntaxique est une contrainte qui force la syntaxe à construire des structures pouvant être identifiées par une entrée lexicale. Le principe de Lexicalisation Exhaustive, donné en (11), peut donc être vu comme le moteur de la dérivation.

(11) Exhaustive Lexicalisation Principle:

Every syntactic feature must be lexicalised (Fábregas 2007)

L'indisponibilité d'une entrée lexicale capable de réaliser une structure peut déclencher certaines opérations de mouvement (*Spell-Out driven movements*, ci-après mouvements

[2] Les représentations phonologiques seront données sous forme orthographique pour simplifier la présentation.

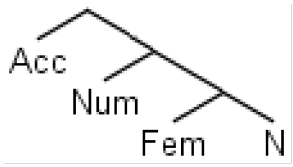
d'évacuation) afin d'obtenir des constituants lexicalisables. En (12), le nom est suivi d'un affixe représentant synthétiquement trois traits fonctionnels.

(12) puell – ās (latin)

fille-Fem.Pl.Acc

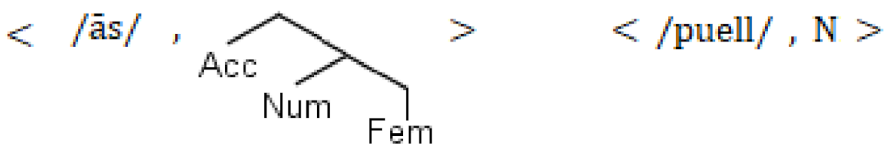
En accord avec la Fseq, les différents traits sont d'abord assemblés dans la syntaxe tel qu'en (13), où N représente la racine nominale, surmontée des traits Féminin (Fém), Pluriel (Num) et Accusatif (Acc).

(13)



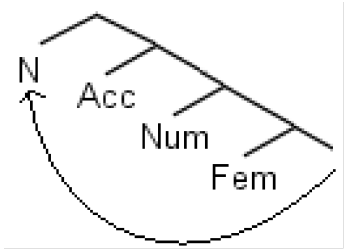
Comme le lexique du latin n'a pas d'entrée lexicale pouvant réaliser entièrement le S-Tree en (13) par une forme supplétive, un mouvement d'évacuation est forcé afin d'obtenir des constituants pouvant être lexicalisés par les entrées en (14).

(14) a. b.



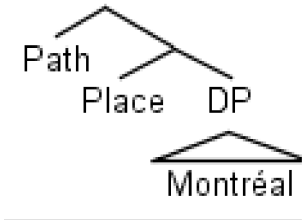
Les mouvements d'évacuation ne laissent pas de copie dans la position d'origine. Toute la partie de l'arbre dominée par la mère de Acc en (15) peut donc être réalisée par (14a) après l'évacuation de N, qui peut quant à lui être réalisé par (14b).

(15)



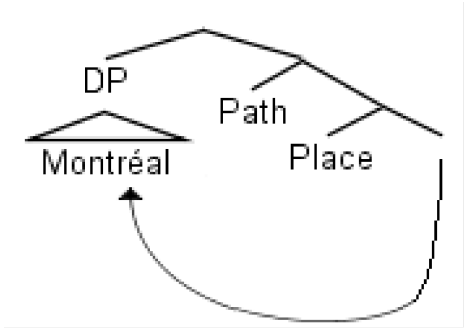
De plus, comme N c-commande asymétriquement les autres traits après le mouvement, le LCA linéarise l'affixe à droite de la racine nominale. L'exemple de *puellās* semble de prime abord faire la prédiction incorrecte que les traits fonctionnels surmontant une catégorie lexicale sont toujours linéarisés à droite. Starke (2018) montre qu'en fait les mécanismes de lexicalisation en NS permettent d'expliquer élégamment la différence entre pré- et post-éléments (affixes ou adpositions) par la forme de leurs L-Trees plutôt qu'en stipulant leur position par un diacritique. Pour voir comment, revenons à l'exemple de la préposition *à* en français. Disons que les traits syntaxiques correspondant à la préposition sont introduits au dessus d'un DP (disons [DP *Montréal*]) tel qu'en (16).

(16)



À l'étape de la dérivation représentée en (16), le constituant DP a été lexicalisé, et la grammaire cherche maintenant à lexicaliser les traits Path et Place afin de respecter la Lexicalisation Exhaustive. Comme il n'y a pas d'entrée lexicale capable de réaliser toute la structure en (16), on s'attendrait, selon la logique illustrée en (13)-(15), à ce que DP soit déplacé au haut de la structure tel qu'en (17) pour permettre de réaliser la structure {Path{Place}} par l'entrée en (10a).

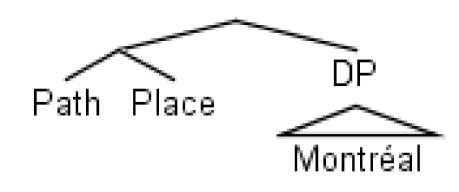
(17)



On prédit alors incorrectement que *à* devrait être une postposition. La raison est que pour que Path et Place forment un constituant, DP doit être évacué en haut de la structure, ce qui implique qu'il soit linéarisé à la gauche de *à*. Starke (2018) suggère que les pré- et

post-éléments ont en fait des structures syntaxiques différentes. Dans le cas des pré-éléments, les traits fonctionnels sont d'abord assemblés dans une dérivation parallèle avant d'entrer dans la dérivation principale, tel qu'en (18).

(18)



En (17), le fait d'avoir d'abord combiné DP avec Place pour ensuite évacuer DP laisse un branchement unaire au-dessus de Place, ce qui n'est pas le cas en (18) parce que Path et Place sont d'abord assemblés entre eux avant de se combiner à DP. La structure de l'adposition est donc {Path,{Place}} en (17) pour une postposition, mais {Path, Place} en (18) pour une préposition. Ce sont donc les entrées lexicales disponibles dans la langue qui forcent la grammaire à assembler les traits dans le bon ordre et qui indirectement dictent une certaine relation de précédence.

(19) L-Trees pour pré- vs post-éléments

a. </à/, {Path, Place}> b. </à/, {Path, {Place}}>

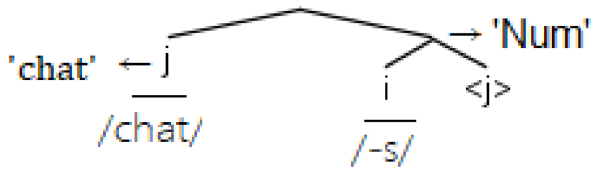
Le fait de dériver l'ordre linéaire par la forme des L-Trees est une stratégie qui sera employée dans le reste de ce travail.

1.4 L'interface sémantique

Étant donné que, dans la perspective OFOH, chaque trait syntaxique doit correspondre à un trait sémantique, l'interface à C-I est plus directe que celle avec S-M : alors que les représentations phonologiques sont obtenues via des entrées lexicales, les représentations sémantiques sont simplement transférées depuis la syntaxe. Toutefois, le fait de concevoir l'interface sémantique comme un transfert porte à confusion. Comme mentionné précédemment, un trait est par définition un élément de représentation pour un module donné. À strictement parler, donc, le « transfert » des traits depuis la syntaxe vers la sémantique reflète la supposition tacite que chaque trait syntaxique correspond à un et un seul trait sémantique. Cette supposition va au-delà de ce qui est justifié du point de vue de la SMT. Le contenu de LEX devrait se limiter à ce qui est nécessaire pour coordonner les bonnes instructions aux deux interfaces, et non être fait sur mesure pour satisfaire l'une des deux interfaces. Le cœur de ma proposition, qui sera développée dans les prochaines sections, est que le contenu sémantique qui est normalement assimilé à des traits syntaxiques est en fait inséré post-syntaxiquement via des entrée lexicales, de la même façon que l'est le contenu phonologique. Prenons par exemple la structure syntaxique en (20)[3]. La partie des traits fonctionnels qui est visible en syntaxe y est explicitement séparée du contenu sémantique.

[3] Comme mentionné précédemment, les mouvements d'évacuation ne laissent pas de copies. Celles-ci seront parfois incluses dans les exemples, mais seulement dans le but de montrer l'historique de la dérivation.

(20)



Les éléments manipulés par la syntaxe ont été réduits à ce qui est strictement nécessaire du point du vue de la SMT, soit des indices à valeur purement différentielle capables d'être distingués par des règles d'insertion comme celles en (21), à chacune des interfaces.

(21) Entrées sémantiques — Entrées phonologiques
a. {i} → ' Num' — b. i → /s/
c. j → 'chat' — d. j → /chat/

L'indice j en (20) est un objet syntaxique qui est réalisé post-syntaxiquement par la forme phonologique /chat/ par la règle (21d), et par la dénotation sémantique 'chat' par la règle (21c). La forme /s/ du pluriel s'obtient quant à elle par la règle (21b) sur l'indice i. Enfin, la règle (21a) associe la dénotation du pluriel à l'objet syntaxique {i}, lequel est obtenu après l'évacuation de j. Le résultat est que la forme /chat/ c-commande asymétriquement /s/, produisant ainsi entre elles la bonne relation de précédence, alors que les deux dénotations se trouvent en position sœur (j'expliquerai au chapitre 3 pourquoi cette configuration est nécessaire pour combiner les deux dénotations par application fonctionnelle). Avant de clore la section, j'aimerais noter certains points importants par rapport à l'approche en (20)-(21). Premièrement, la règle (21a) montre que le contenu sémantique peut être inséré dans un noeud non terminal, exactement comme le matériel phonologique en NS (autrement dit, le système proposé ici étend le PSO à l'insertion sémantique). Deuxièmement, la structure syntaxique en

(20) ne contient pas, à proprement parler, un trait syntaxique PLURIEL, mais plutôt un objet syntaxique réalisé /s/ par la règle (21b) d'une part, et d'autre part un objet syntaxique réalisé 'Num' par la règle (21a). Le dernier point à noter est que, de façon analogue à ce que propose Starke (2018) pour les préfixes/suffixes, la forme des entrées lexicales disponibles est capable à elle seule de forcer la syntaxe à effectuer la bonne séquence d'opérations et indirectement produire une certaine relation de précédence.

Dans le chapitre 2, je vais apporter des arguments en faveur de l'insertion sémantique tardive telle qu'illustrée en (20)-(21) provenant de la morphologie nominale du français, en particulier l'apophonie nominale au pluriel (e.g. *cheval* ~ *chevaux*). Le chapitre 3 soulèvera certains problèmes liés à l'insertion sémantique tardive en ce qui a trait à la Fseq en tant que contrainte sur l'opération Merge; les solutions proposées permettront ensuite d'analyser au chapitre 4 les cas plus complexes de l'infixation et du ablaut.

Chapitre 2 L'idéal concaténatif

2.1 Supplétion et apophonie

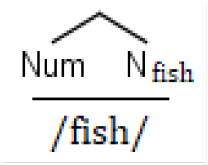
Dans cette section, je vais montrer que l'architecture de la nanosyntaxe (OFOH + PSO phonologique) est trop rigide pour rendre compte de certains processus morphologiques non concaténatifs, puis je montrerai comment l'insertion sémantique tardive telle qu'illustrée en (20) au chapitre précédent apporte une solution à ce problème sans complexifier les mécanismes d'insertion lexicale. Nous avons vu comment l'insertion de contenu phonologique dans des nœuds non terminaux permet d'expliquer simplement les cas où plusieurs éléments syntaxiques sont exprimés par une même forme morphologiquement indivisible. Par exemple, la disponibilité en (22a) d'une entrée lexicale supplétive pour le nom *fish* au pluriel permet de lexicaliser d'un coup la structure en (23), contrairement à (24) où le nom doit être évacué pour permettre de réaliser le trait pluriel séparément.

(22) a. </fish/, {N_{fish}, Num}>

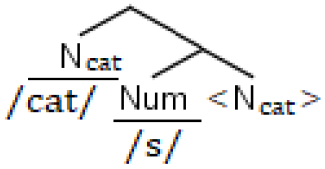
b. </s/, {Num}>

c. </cat/, N_{cat}>

(23)



(24)



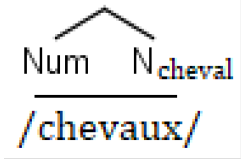
Voyons maintenant comment la même approche se comporte en ce qui concerne l'apophonie en (26), où la forme du nom singulier *cheval* est partiellement modifiée en présence du pluriel.

(25) a. $N_{cheval} \rightarrow$ /cheval/

b. {Num, N_{cheval}} → /chevaux/

Le constituant {Num, N_{cheval}} est ici lexicalisé directement par la forme supplétive *chevaux*.

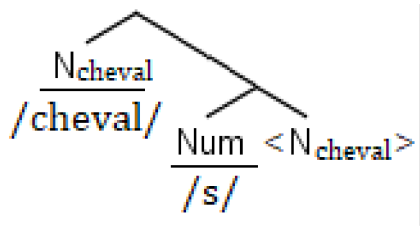
(26)



Le problème avec cette approche est qu'elle ne rend pas compte du fait que la forme supplétive en (25b) conserve une partie de la forme en (25a) (/chev/). Autrement dit, en qui concerne l'analyse en (25)-(26), la forme supplétive en (25b) n'a aucun lien avec celle en (25a); elle pourrait tout aussi bien être /souris/ ; le fait qu'une partie du contenu phonologique de la forme singulière est conservée dans la forme plurielle est traité comme accidentel. En revanche, on

peut voir en (27) que si N est évacué comme en (24), alors on ne peut pas du tout rendre compte du changement de forme en présence du trait pluriel.

(27)



Le problème vient de la rigidité des mécanismes que nous avons admis jusqu'à maintenant pour la lexicalisation. Nous avons seulement deux mécanismes très simples pour réaliser phonologiquement une structure syntaxique : l'insertion lexicale (PSO), qui consiste à remplacer entièrement un constituant par une forme phonologique (c.-à-d. une supplétion), et l'Axiome de correspondance linéaire (LCA) (Kayne 1994), qui assure la linéarisation entre les différentes formes obtenues par insertion lexicale (c.-à-d. leur concaténation). Ces deux mécanismes dérivent ce que Bye et Svenonius (2010) appellent l'Idéal Concaténatif (IC) :

(28) Idéal Concaténatif

a. *Précédence propre* : les morphèmes sont ordonnés linéairement (pas de "chevauchement").
b. *Contiguïté* : les morphèmes sont linéairement contigus (e.g. pas d'infixes ou de circonfixes).
c. *Additivité* : la réalisation peut seulement ajouter du contenu segmental (pas de soustraction).
d. *Préservation* : la réalisation ne peut pas altérer celle d'un autre morphème.
e. *Autonomie segmentale* : le contenu segmental est non contextuel.
f. *Disjonctivité* : pas d'haplologie.

La simplicité de l'IC mérite qu'on tente de le préserver autant que possible (c.-à-d. de limiter le spell-out au PSO et au LCA). Pour ce faire, on peut poser la question suivante : à quoi doit ressembler la syntaxe pour que l'IC soit préservé? Notons tout d'abord que si le concept de morphème en (28) est assimilé aux nœuds terminaux de la syntaxe (autrement dit, si on adopte la position classique selon laquelle l'insertion de matériel phonologique est limitée aux nœuds terminaux), il est évident que les faits attestés comportent des écarts avec l'IC. Les langues naturelles montrent systématiquement des situations où un même trait syntaxique est réalisé par différentes formes phonologiques en fonction de son environnement. Par exemple, si on veut analyser (23)-(24) en limitant l'insertion aux nœuds terminaux, on est forcé de postuler deux allomorphes pour le trait Num, /s/ et zéro, puis de spécifier les contextes où doit apparaître chaque allomorphe.

(29) Num → Ø / _ [N_{fish}

Num → /-s/ Elsewhere

L'analyse en (29), typique par exemple en Morphologie Distribuée (DM) (voir entre autres Harley et Noyer 1999; Bobaljik 2012), revient à admettre des écarts avec l'IC, en particulier (28d-e), ce qu'on cherche ici à éviter. L'analyse en (23), en retirant la supposition tacite que l'insertion est limitée aux nœuds terminaux, permet en revanche de limiter l'insertion lexicale à un format non contextuel (comparer (4a-b), Chapitre 1) de sorte que chaque objet syntaxique est associé à une seule forme : /s/ est la réalisation de Num, alors que /fish/ est la réalisation de {Num, N_{fish}}. Le PSO s'avère donc un moyen de simplifier l'interface phonologique et de préserver l'IC dans le cas de (23). Pour le problème en (26)-(27), cependant, le PSO ne peut

pas nous aider parce que l'alternance morphologique (*al* ~ *aux*) ne cible pas un constituant syntaxique mais seulement une partie de la racine nominale. Pour conserver l'IC, le seul moyen de cibler isolément le matériel apophonique serait de raffiner la structure syntaxique, mais le postulat selon lequel les traits syntaxiques sont en relation 1:1 avec les traits sémantiques (OFOH) nous interdit de diviser N_{cheval} en plusieurs nœuds. Je vais maintenant montrer que la difficulté que pose l'apophonie par rapport à l'IC vient en fait de la borne imposée par la compositionnalité sémantique sur la structure syntaxique, puis je vais suggérer que le fait d'adopter le PSO pour le contenu sémantique permet d'abandonner cette limitation sans conséquence pour le fonctionnement de la syntaxe. D'abord, comparons l'alternance *cheval/aux* avec celle en (30c-d)

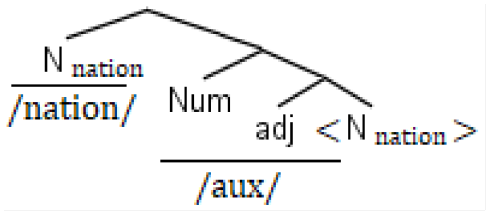
(30) a. cheval b. chevaux

c. national d. nationaux

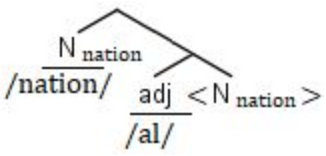
Malgré la similarité formelle, l'alternance entre (30c-d) se laisse analyser beaucoup plus aisément à l'intérieur des mécanismes de la nanosyntaxe.

(31)

a.

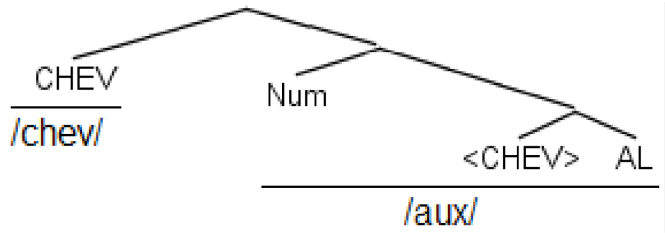


b.



Le fait d'assembler dans la syntaxe la racine nominale et le suffixe *-al* n'est pas controversé dans un cadre comme la nanosyntaxe; *-al* est un suffixe dérivationnel productif en français, et la nanosyntaxe ne fait pas de distinction entre la morphologie dérivationnelle et flexionnelle : tous les traits sont combinés par Merge. La facilité d'analyser (30c-d) par rapport à (30a-b) vient du fait que le matériel supplétif correspond à un affixe avec un apport sémantique identifiable plutôt qu'à une partie de la racine nominale, laquelle reste syntaxiquement atomique. Pour appliquer le même traitement qu'en (31) au cas de *chevaux*, il faudrait scinder la racine nominale en (au moins) deux éléments syntaxiques, un geste beaucoup plus lourd de conséquences :

(32)



La raison pour laquelle l'analyse en (32) semble à première vue inconcevable est que la segmentation de la racine en deux éléments syntaxiques ne semble pas justifiée du point de vue de la sémantique. Les éléments CHEV et AL ne sont pas chacun associés à un contenu sémantique comme le sont N_{nation} et Adj en (31). La compositionnalité sémantique impose donc une borne inférieure à la granularité de la structure syntaxique, et cette borne découle du postulat que chaque élément syntaxique doit correspondre à une et une seule dénotation en sémantique. Cette contrainte rappelle en fait la définition classique du morphème dans la

tradition structuraliste, comme unité *isolable* et *signifiante*. La segmentation de la racine en (32) viole le critère de signifiance.

(33) Fábregas & Scalise (2012) : 4 critères pour le morphème

a. **Signifiance**

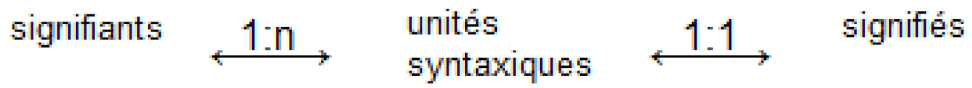
b. Distinctivité

c. Récurrence

d. Isolabilité

Notons qu'en NS, où on admet l'insertion de matériel phonologique dans des nœuds non terminaux, il n'y a pas de borne analogue imposée par l'isolabilité. Autrement dit, on admet qu'un constituant composé de plusieurs éléments syntaxiques puisse être réalisé par une seule forme morphologiquement indivisible, mais pas par une seule unité sémantique indivisible. En d'autres termes, la NS libère les éléments syntaxiques de l'obligation de respecter le critère d'isolabilité, mais pas celui de signifiance. La situation est schématisée en (34), où une forme phonologique peut correspondre à un ou plusieurs éléments syntaxiques, alors que chaque élément sémantique doit correspondre à un seul élément syntaxique et vice versa.

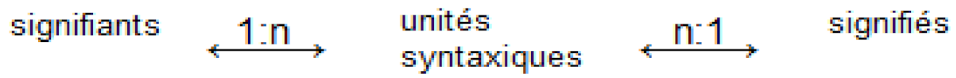
(34)



L'insertion tardive du contenu phonologique et le PSO ont pour effet d'évacuer de la syntaxe le critère d'isolabilité : l'isolabilité segmentale est un critère qui s'applique après spell-out, pas sur

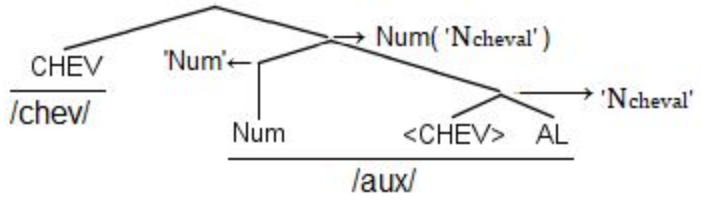
les unités de la structure syntaxique. Inversement, une façon de voir la stipulation d'une correspondance 1:1 entre traits syntaxiques et sémantiques est que l'insertion sémantique est limitée aux nœuds terminaux. Ma proposition est en somme d'évacuer le critère de signifiance de la syntaxe de façon analogue au critère d'isolabilité en admettant l'insertion tardive et le PSO pour le contenu sémantique. Le portrait en (34) est ainsi remplacé par celui en (35), où chaque élément de la représentation sémantique peut correspondre à un ou plusieurs éléments dans la structure syntaxique.

(35)



En (36), CHEV et AL représentent des indices sans contenu phonologique ni sémantique. Ces indices, comme tous les éléments syntaxiques, n'ont pour fonction que celle d'être identifiables par des règles lexicales afin de fournir des instructions à C-I et S-M.

(36)



Le rôle de la syntaxe est d'assembler ces indices en constituants pouvant être lexicalisés à chacune des interfaces par des entrées comme en (37).

(37) Entrées phonologiques

a. <CHEV → /chev/>

b. <{AL} → /al/>

c. <NUM → /s/ >

d. <{{NUM},{AL}} → /o/>

Entrées sémantiques

e. {NUM}→ 'Num'

f. {AL, CHEV} → 'Ncheval'

g. AL → 'Adj'

La flèche la plus à droite en (36) indique que le contenu sémantique correspondant à Ncheval est la réalisation du constituant {CHEV, AL} formé à partir de ces deux indices (obtenue par la règle (37f)), plutôt que celle de l'un ou l'autre de ces indices. Une fois cette interprétation sémantique obtenue, Num est combiné à l'objet syntaxique {CHEV, AL}, après quoi la dénotation 'Num' (obtenue par (37e)) se combine sémantiquement avec 'Ncheval' par application fonctionnelle. La dénotation du constituant {{Num},{CHEV, AL}} est l'output de cette application, et est donc égal à 'Num(Ncheval)'. Une fois cette dénotation obtenue, l'indice CHEV est libre d'être évacué pour permettre de réaliser la forme supplétive de {{Num},{AL}} par la règle (37d), alors que CHEV est réalisé par (37a)[4]. L'apophonie équivaut donc à une supplétion visant un élément plus petit que la racine.

La clé de cette analyse est que le matériel sémantique est inséré de manière parallèle au matériel phonologique. Le PSO s'applique maintenant aussi pour l'insertion du contenu

[4] Je laisse pour l'instant de côté la question de savoir comment le noeud au sommet de (36) hérite de la dénotation de son nœud fille (obtenue par l'application Num(Ncheval)) après le déplacement de CHEV. Ce problème sera résolu au chapitre 4.

sémantique, ce qui a pour effet de briser la relation 1:1 entre les éléments syntaxiques et sémantiques. On explique ainsi que (30c-d) montre la même alternance que (30a-b) malgré la contribution sémantique variable de la forme *-al*/*-aux*. Dans les deux cas, il s'agit du même élément syntaxique, pouvant être identifié par les règles phonologiques (37b) et (37d). La différence est qu'en (30b), cet indice est à l'intérieur d'une structure idiomatique, c'est-à-dire une structure interprétée directement via une entrée lexicale; le contenu sémantique obtenu par (37g) est donc remplacé par celui obtenu par (37f). Notez que même si les éléments syntaxiques sont dénués de contenu sémantique, la structure finale doit recevoir une interprétation à chacune des interfaces.

(38) Lexicalisation Exhaustive (révisé): la structure syntaxique doit être lexicalisée *phonologiquement et sémantiquement*.

Pourvu que (38) soit satisfait, la structure syntaxique peut être interprétée de façon non simultanée à chacune des interfaces, ce qui permet des décalages comme en (36), où l'indice AL est réalisé à l'interface sémantique au sein du constituant {CHEV, AL}, mais au sein du constituant {{Num},{AL}} à l'interface phonologique. L'important est que chacune des interfaces reçoit des instructions qu'elle peut interpréter selon ses propres conditions.

2.2 Affixes idiomatiques

Je vais maintenant porter attention à un fait de la morphologie du français qui appuie l'hypothèse qu'une racine peut correspondre à un objet syntaxique complexe. On trouve en

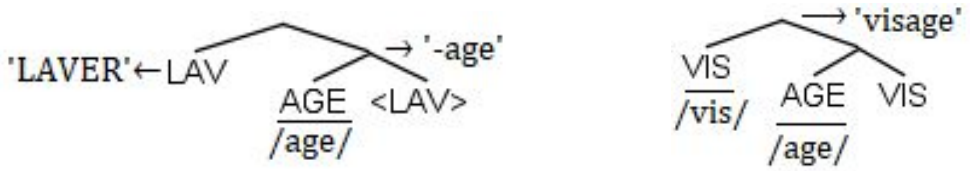
français un grand nombre d'affixes dérivationnels dont la forme apparaît également à l'intérieur de racines. En voici quelques exemples.

(39)	a. Dérivationnel	b. Idiomatique
	i. lavage, collage	i. visage, orage, ménage
	ii. fillette, maisonnette	ii. sornette, alouette
	iii.garçonnet, cochonnet,	iii. palet, briquet, sobriquet, gourmet
	iv. motard, communard	iv. renard, canard, corbillard, babillard,
	v. national	v. cheval, fanal

La série en (39b) montre des terminaisons de noms qui correspondent phonologiquement mais non sémantiquement à celles en (39a). J'appellerai désormais les terminaisons telles que celles en (39b) des « affixes idiomatiques ». La raison pour laquelle ces formes constituent un argument en faveur de la présente analyse est simple : si on suppose une relation 1:1 entre éléments syntaxiques et sémantiques, alors on doit considérer les noms en (39b) comme monomorphémiques parce que leur sens n'est pas décomposable sémantiquement. Or, si les noms en (39b) sont la réalisation d'un seul élément syntaxique, il apparaît étonnant que les mêmes chaînes de phonèmes reviennent d'une réalisation à l'autre, qui plus est des chaînes qui correspondent aussi à des suffixes dérivationnels comme en (39a). On s'attendrait plutôt à trouver en (39b) des terminaisons qui varient aléatoirement à l'intérieur de ce que permet la phonotactique du français. Si en revanche les racines sont correspondent à des structures syntaxiques composées de plusieurs éléments et interprétées par insertion sémantique, il est normal qu'on retrouve les mêmes éléments dans plusieurs racines puisque celles-ci sont aussi

des constructions syntaxiques. Le fait qu'une même chaîne segmentale, par exemple *-age* en (40), ait une contribution sémantique variable (tantôt compositionnelle comme en (40b), tantôt idiomatique comme en (40a)) est une conséquence naturelle de notre architecture, puisque l'insertion du contenu sémantique peut avoir lieu sur le constituant {AGE} par la règle (41e) (auquel cas l'affixe est dérivationnel) ou sur le constituant {VIS, {AGE}} par la règle (41f) (auquel cas l'affixe est idiomatique). Dans les deux cas, l'élément AGE est identifié par les mêmes entrées phonologiques; la forme phonologique /age/ est obtenue par la règle (41c).

(40) a. b.



(41) a. </vis/, VIS>
b. </lav/, LAV>
c. </age/, AGE>
d. <LAV, 'LAVER'>
e. <{AGE}, '-age'>
f. <{VIS,{AGE}}, 'VISAGE'>

Le genre féminin peut être vu comme un autre exemple d'affixe idiomatique; il fonctionne tour à tour comme un marqueur de classe sans contribution sémantique apparente (genre grammatical) et comme affixe dérivationnel contribuant une information sur le sexe du référent (genre sémantique). Sa réalisation phonologique, en revanche, demeure stable, sous forme d'un schwa dont l'effet est de saigner la chute des consonnes finales. Pour illustrer ceci, prenons le fragment de lexique en (42). Je rappelle que les éléments à gauche de la virgule

sont des indices dont la valeur est strictement différentielle. Leur nom a seulement pour fonction de faciliter l'illustration.

(42) Entrées phonologiques Entrées sémantiques

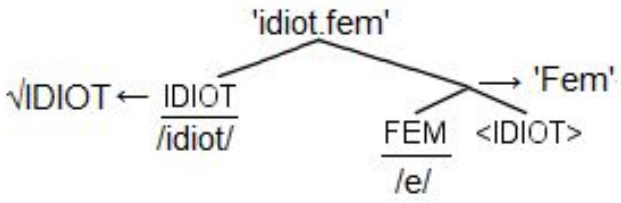
a. <FEM, /e/> d. <{FEM}, 'Fem'>

b. <CHAIS, /chais/> e. <{CHAIS,{FEM}} , 'chaise'>

c. <IDIOT, /idiot/> f. <IDIOT, 'idiot'>

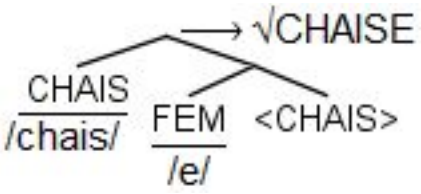
En (43), {FEM} et IDIOT sont interprétés séparément par (42d) et (42f) respectivement, après quoi ils sont combinés sémantiquement pas application fonctionnelle. Les formes obtenues par les règles (42c) et (42a) sont quant à elles dans une relation de c-commande asymétrique, ce qui permet de linéariser la structure.

(43)



En (44), l'objet syntaxique {CHAIS,{FEM}} est interprété directement par l'entrée en (42e). Comme en (43), il y a une c-commande asymétrique entre les deux formes.

(44)



L'analyse du genre comme un affixe idiomatique permet de rendre compte d'une part de la stabilité de sa réalisation phonologique et d'autre part de la variabilité de sa contribution sémantique, tour à tour comme un marqueur de classe et comme un affixe dérivationnel. Pour clore cette section, je vais illustrer un problème posé par une interaction entre le genre grammatical et le suffixe diminutif *-et(te)*. Ce suffixe est productif et peut être inséré après un nom X pour donner approximativement le sens de 'petit X'. Lorsque le nom affixé est grammaticalement féminin, le suffixe diminutif prend la marque du féminin (*-ette*, avec le /t/ final audible). Lorsque le nom est masculin, le suffixe diminutif prend la forme masculine (*-et*, avec chute du /t/ final)[5]. Des exemples sont donnés en (45).

(45) a. une chaise d. un cochon

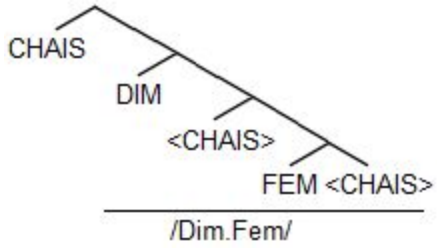
b. une chaisette e. un cochonnet

Dans la présente approche, pour dériver (45b), les éléments syntaxiques FÉM et CHAIS doivent dans un premier temps former un constituant pour obtenir l'interprétation idiomatique ‘chaise’ (autrement dit, il y a une étape dans la dérivation de (45b) qui est identique à (44)). On sait aussi qu’à une étape ultérieure, FÉM doit se trouver derrière le diminutif, puisque c'est

[5] Merci à Mireille Tremblay de m’avoir signalé ce fait.

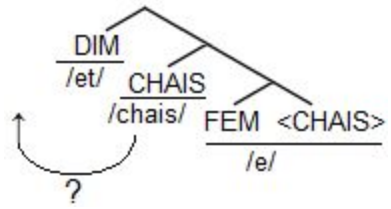
le schwa du féminin qui saigne la chute du /t/ final pour donner la forme féminine du suffixe (*-ette*). Le problème est qu'il est impossible de dériver les trois morphèmes dans cet ordre (CHAIS-DIM-FÉM). Voici pourquoi : à l'étape correspondant à (44), il y a deux façons dont DIM peut être introduit dans la dérivation; la première est illustrée en (46) : DIM est combiné à l'arbre de (44) sans avoir été préalablement prononcé, ce qui déclenche l'évacuation de CHAIS. Le problème est alors que soit DIM et FEM devraient se réaliser par une forme synthétique (c'est-à-dire qu'on la gloserait Dim.Fém plutôt que Dim-Fém) comme en (46), soit DIM devrait se réaliser comme un préfixe.

(46)



La deuxième possibilité est que l'affixe diminutif soit réalisé phonologiquement avant d'être introduit dans la dérivation principale (voir (18)-(19), Chapitre 1), comme en (47). Le problème ici est qu'il ne reste rien dans la structure qui n'a pas été prononcé, et le mouvement de CHAIS ne peut donc pas être déclenché (je rappelle que les mouvements responsables de linéariser les affixes fonctionnels sont déclenchés par la nécessité de réaliser toute la structure (Lexicalisation exhaustive)).

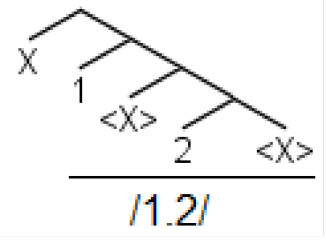
(47)



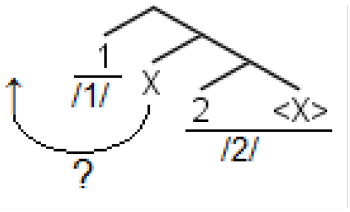
En fait, ce problème survient chaque fois qu'on trouve les suffixes X, Y où X précède Y et X est plus élevé que Y dans la Fseq. En d'autres mots, le système actuel ne peut pas dériver de suffixes « décroissants », c'est-à-dire des séquences de suffixes dans lesquelles les suffixes linéairement plus près de la racine sont plus élevés dans la Fseq. La situation en (46)-(47) est donc une instance d'un problème plus général illustré en (48), où X représente la racine, et 1 et 2 représentent des affixes fonctionnels (1 plus élevé que 2 dans la Fseq)[6].

(48) Problème des suffixes décroissants

a. Suffixe supplétif



b. Évacuation impossible



Le problème des suffixes décroissants est indépendant de la présente proposition puisque plusieurs ordres attestés par Cinque (2005) (voir Chapitre 3) sont indérivables dans le système

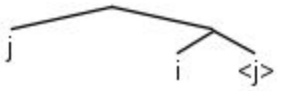
[6] Le Mirror Principle (Baker 1988) prédit que les suffixes devraient toujours être “croissants”. Dans le présent système (de même qu’en nanosyntaxe), la distinction entre un affixe et un morphème libre (un mot) est informulable dans la syntaxe, donc le problème en (48) demeure.

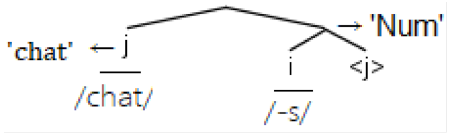
nanosyntaxique. Plusieurs des ordres dérivés par Cinque nécessitent en effet un mouvement cyclique successif de la racine nominale à travers une série de suffixes fonctionnels. Dans le présent cadre, le problème s'applique aussi aux apophonies centrales comme le ablaut (e.g. *mouse*/*mice*) et aux infixes. Pour voir comment, il suffit d'imaginer que 2 et X sont deux morceaux de la racine (structure idiomatique), et que 1 est un trait fonctionnel devant se réaliser comme infixe. Dans le prochaine chapitre, je vais me pencher sur un problème indépendant que pose l'insertion sémantique tardive pour l'application de la Fseq telle que formulée en (3). Les solutions apportées vont aussi permettre de résoudre le problème des suffixes décroissants, et par le fait même celui de l'infixation et du ablaut.

Chapitre 3 Free Merge : dériver la Fseq en sémantique

La Fseq contraint l'ordre dans lequel la syntaxe combine les traits fonctionnels en faisant référence à leur contenu sémantique. Or, dans ma proposition, le contenu sémantique n'est pas accessible à la syntaxe. Par exemple, en (49a) la syntaxe n'a aucune idée de la dénotation qui sera éventuellement attribuée aux constituants en (49b). La seule information qui est accessible la syntaxe est la configuration dans laquelle se trouvent les indices, ce qui est, comme nous avons vu, l'information minimalement nécessaire pour appliquer des entrées lexicales à chacune des interfaces. Il est donc impossible de contraindre Merge en faisant référence au contenu sémantique qui n'a pas encore été inséré.

(49) a. b.





L'architecture que je propose nécessite donc de trouver une alternative à la Fseq pour expliquer la distribution des traits fonctionnels. Du point de vue de la SMT, on aurait par ailleurs avantage à dériver la Fh par des facteurs externes comme mentionné au chapitre 1. Dans ce chapitre, je vais montrer comment la Fseq peut être réinterprétée en termes de types sémantiques plutôt qu'en termes de catégories syntaxiques. Cette approche permet de laisser Merge s'appliquer librement; les dérivations qui contreviennent à la Hiérarchie Fonctionnelle

sont celles qui seront bloquées par les systèmes externes. Je vais commencer par illustrer comment Cinque (2005) utilise la Fseq pour dériver la distribution de certaines catégories du syntagme nominal à travers les langues, puis j'introduirai mon approche en montrant qu'elle permet de dériver la même généralisation sans avoir recours à la Fseq.

Cinque (2005) observe que, parmi 24 ordres mathématiquement possibles entre 4 catégories du syntagme nominal (Démonstratifs, Numéraux, Adjectifs et Noms), seulement 14 sont attestés à travers les langues (voir plus loin le Tableau 1). Sa généralisation est un raffinement de l'Universel 20 de Greenberg (1963). Il montre ensuite qu'il est possible de dériver tous et seulement les ordres attestés à l'aide des trois propositions en (50).

(50)

a. Fseq = < ..., Dem, Num, Adj, N, ... >

b. Restriction universelle sur le mouvement: les mouvements (non reliés au focus) doivent cibler des syntagmes contenant l'élément de classe ouverte (la racine) (viz. N).

c. L'axiome de correspondance linéaire (Kayne 1994)

L'approche de Cinque consiste donc en un premier temps à stipuler un ordre de base universel selon lequel les traits sont assemblés par Merge (la Fseq) puis à définir la classe des mouvements pouvant modifier cet ordre. Les différentes structures résultant de ces mouvements donnent ensuite différents ordres à spell-out selon le LCA. Pour éviter de

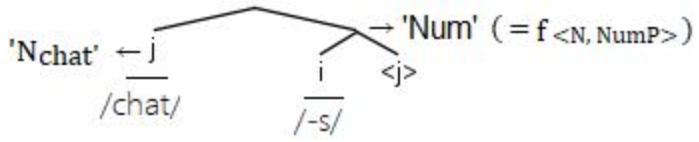
contraindre directement l'ordre d'application de Merge, nous devrons le faire indirectement en exploitant les conditions imposées par les interfaces sur l'output de Merge. Mon approche consiste à définir comment la structure syntaxique interagit avec chacune des interfaces de façon à ce que l'ensemble des ordres attestés corresponde exactement à l'ensemble des dérivations syntaxiques qui satisfont à la fois aux conditions de C-I et de S-M. Dans un premier temps, faisons l'inventaire des conditions d'interfaces que nous avons jusqu'à maintenant.

Du côté de l'interface phonologique, nous avons déterminé qu'elle imposait essentiellement deux contraintes : la syntaxe doit construire des objets pouvant être identifiés par des entrées lexicales, et établir entre les réalisations obtenues des asymétries pouvant être interprétées comme des relations de précédence. Du côté de la sémantique, nous avons proposé que la syntaxe devait aussi permettre l'application de règles lexicales, de façon similaire et parallèle aux règles lexicales phonologiques, afin d'obtenir les représentations à la base du calcul sémantique. Toutefois, nous n'avons pas encore abordé la façon dont ces représentations sont ensuite combinées par C-I pour obtenir le sens de la phrase. Je propose que la combinaison des représentations sémantiques obtenues par insertion lexicale exige que la syntaxe établisse entre elles certaines configurations, de la même façon que le fait la linéarisation via le LCA.

Supposons que la principale opération pour le calcul sémantique est l'application fonctionnelle (Heim & Kratzer 1998). La dénotation d'un trait fonctionnel est donc égale à une fonction prenant comme argument une autre dénotation présente dans la représentation sémantique. Je

propose qu'il existe une certaine configuration syntaxique que l'interface à C-I interprète comme l'instruction de combiner deux dénotations par application fonctionnelle, de la même façon que l'interface phonologique interprète une certaine configuration (la c-commande asymétrique) comme l'instruction de combiner deux représentations phonologiques par concaténation. Supposons que l'application fonctionnelle est réalisée entre nœuds sœurs.

(51)

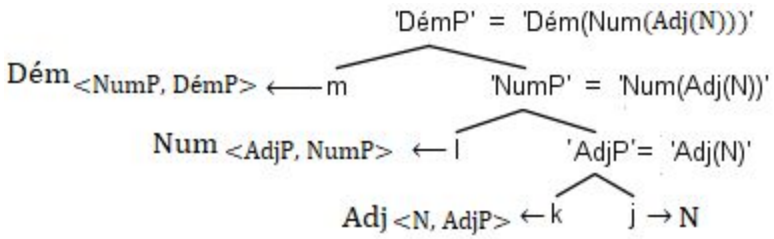


La dénotation 'Num', obtenue par insertion sur le nœud {i}, est égale à une fonction de type <N, NumP>, c'est-à-dire une fonction demandant un argument de type N et renvoyant un output d'un certain type, que j'appelle ici NumP pour simplifier, bien qu'il ne s'agit pas d'une projection de Num au sens classique du terme. La dénotation 'Nchat' est quant à elle obtenue par insertion lexicale sur le noeud j. Comme ces deux dénotations sont insérées sur des nœuds sœurs, cette configuration est interprétée comme l'instruction de les combiner par application, et comme 'Nchat' fait partie du domaine de la fonction 'Num', l'opération est en mesure de fournir un output. La dénotation de l'arbre en (51) devient donc 'Num(Nchat)', de type NumP. Si en revanche les dénotations étaient telles qu'il est impossible d'appliquer l'une à l'autre, alors l'opération ne serait pas en mesure de renvoyer un output et la dérivation échouerait.

La conclusion est que si l'application fonctionnelle est réalisée dans une certaine configuration

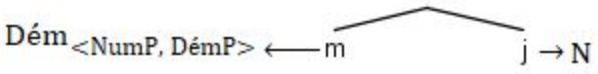
syntaxique, alors les éléments syntaxiques sont forcés d'être combinés et lexicalisés dans un ordre précis pour éviter une incompatibilité de type. Le fait que le pluriel « sélectionne » un nom est déjà pris en charge par leurs dénotations, et exprimer la même relation par la Fseq est donc redondant. En suivant cette logique, le fragment de Fseq en (50a) peut être réinterprété de la façon suivante. Chacun des traits en (50a) correspond en (52) à une fonction (insérée tardivement) dont le domaine correspond à l'output de la fonction précédente.

(52)



L'arbre (52) correspond à une situation où les quatre dénotations obtenues par insertion sémantique (correspondant aux quatre traits dans la Fseq de Cinque) sont toutes présentes dans la dérivation, et où les fonctions s'appliquent l'une après l'autre, l'output de l'une devenant l'input de la suivante. Il y a deux problèmes avec cette situation. Premièrement, certaines catégories sont facultatives. Par exemple, en essayant de dériver le DP *ce chat*, 'N' ne semble pas être du bon type pour être appliqué à 'Dém'.

(53) *Ce chat*

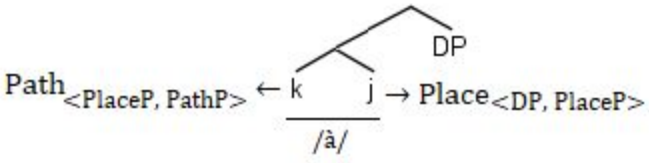


Une solution potentielle serait que les types NumP, AdjP et N sont en relation d'inclusion.

(54) NumP ⊇ AdjP ⊇ N

Si N est un sous-type de NumP, alors le nom en (53) fait partie du domaine de la fonction 'Dém'. On peut ainsi compte du fait que Num peut prendre directement N comme argument, et du fait que N ne peut pas être combiné à Adj après Num, parce que NumP ne fait pas partie du domaine de Adj. Le deuxième problème est que, comme nous l'avons vu au chapitre 1, la dérivation d'éléments préfixaux nécessite que certains traits fonctionnels soient assemblés dans une dérivation parallèle avant d'être ajoutés à la structure principale de la phrase (voir (16)-(19)).

(55)



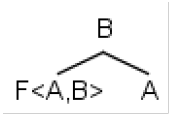
Pour satisfaire l'entrée lexicale en (56), Place doit être combiné à Path avant d'être combiné à

DP. Une fois avoir assemblé le constituant {Path, Place}, les traits fonctionnels sont combinés à DP. Or, ni 'Path' ni 'Place' ne constitue un argument adéquat pour l'autre fonction.

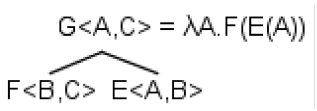
(56) Entrée lexicale: </à/, {Path,Place}>

Le noeud syntaxique {Path, Place} n'est donc pas en mesure de recevoir une dénotation. Pour permettre aux traits préfixaux comme en (55) de retarder leur saturation, je vais postuler en plus de l'application fonctionnelle une deuxième opération sémantique : la Composition fonctionnelle (58) est une opération qui combine deux fonctions pour donner une nouvelle fonction qui équivaut à appliquer les deux premières fonctions à la chaîne (voir Steedman 1996, ou l'ensemble de la littérature en Combinatory Categorial Grammar (CCG), pour une opération similaire)[7].

(57) Application Fonctionnelle



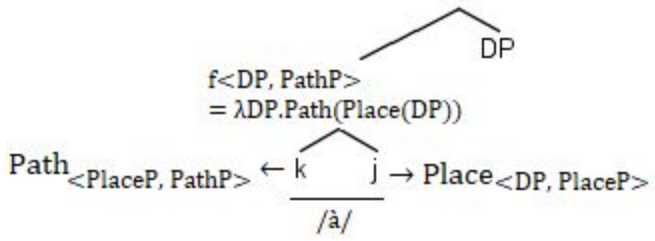
(58) Composition Fonctionnelle



L'opération en (58) a pour effet de « fusionner », plutôt que d'appliquer l'une à l'autre, les deux fonctions en (55). La dénotation de {Path, Place} devient donc une fonction f, de type <DP,PathP>, équivalente à l'application successive Place et Path.

[7] Nous allons provisoirement supposer que la composition fonctionnelle est aussi réalisée en relation sœur. Des ajustements seront apportés dans la prochaine section.

(59)



Dans ce cadre, la généralisation de Cinque (2005) peut être dérivée par la contrainte suivante, dispensant de la nécessité d'une Fseq:

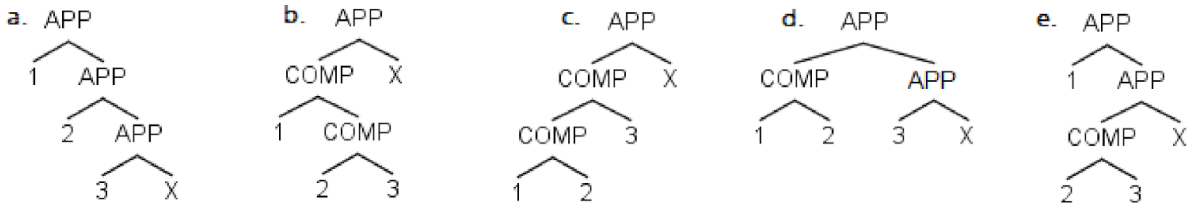
(60) a. Si les fonctions f<A,B> et g<B,C> se combinent par Composition Fonctionnelle pour donner la fonction h<A,C> = λA.g(f(A)), alors la réalisation de g doit précéder celle de f.

En d'autres mots, (60) dit que si deux fonctions sont combinées par composition, la relation de précédence entre les formes phonologiques correspondant à ces fonctions doit obligatoirement être telle que la fonction qui est hiérarchiquement plus élevée (celle qui s'applique en deuxième) précède l'autre. Par exemple, si Path et Place sont combinés par composition, Path doit précéder Place, parce que Path doit s'appliquer à l'output de Place et non l'inverse. Ce qui reste implicite en (60) est que quand deux dénotations se combinent par Application Fonctionnelle, les deux ordres sont possibles à travers les langues. Par exemple, si 'Num' prend 'N' comme argument, les deux ordres sont en principe possibles[8].

[8] Il y a une certaine similarité entre mon approche et celle de Abels & Neeleman (2009), dans le sens qu'elle permet une certaine flexibilité dans l'ordre entre têtes et compléments. Leur approche demeure toutefois fondamentalement différente de la mienne, et plus proche de Cinque (2005), en ce qu'elle conserve la Fseq et l'idée d'un ordre de base suivi de mouvements.

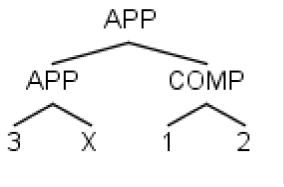
La contrainte en (60) dérive exhaustivement et exclusivement les ordres attestés par Cinque. Une conséquence intéressante est que le nombre de dérivations possibles pour chaque ordre est sujet à variation. À titre d'exemple, les cinq dérivations possibles pour l'ordre Dém-Num-Adj-N (1-2-3-X) sont illustrées en (61), alors que l'unique dérivation possible pour l'ordre Adj-N-Dém-Num (3-X-1-2) est montrée en (62). Les noeuds identifiés "APP" sont ceux dont les noeuds filles se combinent par application fonctionnelle, et ceux identifiés "COMP" par composition fonctionnelle.

(61)



La contrainte (60) se manifeste dans le fait que pour toutes les paires de nœuds qui sont dominées par une étiquette COMP, représentant deux fonctions se combinant par composition, le nœud de droite a toujours un nombre plus élevé (c'est-à-dire que la fonction hiérarchiquement plus élevée doit être à gauche). Il y a cinq façon d'arriver à l'ordre 123X en respectant cette contrainte, mais une seule façon d'arriver à 3X12.

(62)



Le Tableau 1 donne un sommaire comparant la fréquence de chaque ordre d'après Cinque (2005) avec le nombre de dérivations possibles dans le présent système.

TABLEAU 1

	Ordre	Fréquence	Nb de dérivations
a.	DEM, NUM, A, N	TRÈS FRÉQUENT	5
b.	DEM, NUM, N, A	FRÉQUENT	2
c.	DEM, N, NUM. A	TRÈS RARE	1
d.	N, DEM, NUM, A	RARE	2
e.	NUM, DEM, A, N	*	0
f.	NUM, DEM, N, A	*	0
g.	NUM, N, DEM, A	*	0
h.	N, NUM, DEM, A	*	0
i.	A, DEM, NUM, N	*	0
j.	A, DEM, N, NUM	*	0
k.	A, N, DEM, NUM	TRÈS RARE	1
l.	N, A, DEM, NUM	RARE	1
m.	DEM, A, NUM, N	*	0
n.	DEM, A, N, NUM	TRÈS RARE	1

o.	DEM, N, A, NUM	FRÉQUENT	1
p.	N, DEM, A, NUM	TRÈS RARE (possiblement *)	0
q.	NUM, A, DEM, N	*	0
r.	NUM, A, N, DEM	TRÈS RARE	2
s.	NUM, N, A, DEM	RARE	1
t.	N, NUM, A DEM	RARE	1
u.	A, NUM, DEM, N	*	0
v.	A, NUM, N, DEM	*	0
w.	A, N, NUM, DEM	TRÈS RARE	1
x.	N, A, NUM, DEM	TRÈS FRÉQUENT	1
		TOTAL	20

La relation entre le nombre de dérivations possibles et la fréquence d'un ordre donné à travers les langues est à ce point-ci incertaine. Les deux variables semblent à première vue corrélées (p. ex. l'ordre Dem-Num-A-N, l'un des deux ordres les plus fréquents d'après Cinque, est celui ayant le plus grand nombre de dérivations avec 5, soit 25 % de toutes les dérivations possibles). L'exception la plus manifeste est l'ordre miroir N-A-Num-Dem, aussi identifié par Cinque comme très fréquent, mais ayant une seule dérivation possible. En faisant abstraction de la fréquence, le Tableau 1 montre crucialement que tous les ordres non attestés (*) ont 0 dérivation possible, alors que tous les ordres attestés ont au moins une dérivation possible. Le seul ordre potentiellement attesté qui est non dérivable par (60) est N-DEM-A-NUM (numéro p du Tableau 1), que Cinque identifie comme possiblement erroné (*spurious*). La validité de

mon approche repose donc sur le caractère erroné de la description donnée pour les très rares langues concernées. Le fait que le seul ordre potentiellement problématique est justement celui qui est mis en doute est toutefois de bon augure.

Chapitre 4 Autres applications

Il reste maintenant à expliquer pourquoi la composition fonctionnelle demande une certaine relation de précédence, contrairement à l'application fonctionnelle (autrement dit.dériver (60)). Je crois que la clé se trouve dans les relations syntaxiques fournissant des instructions à chacune des interfaces. D'une part, l'interface à S-M interprète une certaine configuration comme une relation de précédence (LCA). D'autre part, l'interface C-I interprète une certaine configuration comme l'instruction de combiner deux fonctions par composition. L'hypothèse la plus simple serait donc que la configuration qui est interprétée en phonologie comme une précédence est la même configuration qui est interprétée en sémantique comme une composition. Cette hypothèse est plausible dans la mesure où la composition fonctionnelle, comme la précédence, est une relation asymétrique. Supposons que (62a-b) sont des instructions reçues de la syntaxe par les interfaces sémantique et phonologique respectivement:

(62) a. COMP(A,B) *Combiner A et B par composition fonctionnelle*

b. PREC(A,B) *A précède B*

Étant donné que ces opérations sont asymétriques (COMP(A,B) ≠ COMP(B,A) et PREC(A,B) ≠ PREC(B,A)), la structure syntaxique doit établir une distinction entre les deux représentations pour que les interfaces interprètent l'instruction correctement.. Comme on suppose déjà que la précédence découle de la c-commande asymétrique (LCA), je propose les deux opérations d'interfaces suivantes :

(63) Concaténation

{A{B}} → PREC(A,B) où A, B sont associés à des représentations phonologiques

(64) Composition

{A{B}} → COMP(A,B) où A, B sont associés à des représentations sémantiques

Quant à l'application fonctionnelle, celle-ci a certes également un caractère asymétrique, mais cette asymétrie réside dans le fait que l'un des objets peut prend l'autre comme argument. L'asymétrie est déjà encodée dans leur dénotation et n'a pas besoin d'être spécifiée par la syntaxe pour que la sémantique puisse les assembler correctement. Je propose donc que l'application fonctionnelle est réalisée dans une configuration symétrique (entre nœuds sœurs).

(65) Application

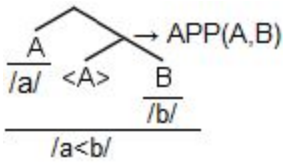
i. {A,B} → APP(A,B) où A, B sont associés à des représentation sémantiques

ii. APP(A,B) = A(B) ou B(A)

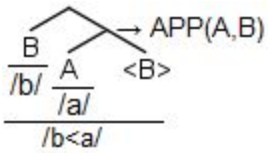
La nécessité de linéariser la structure via (63) force ensuite l'évacuation d'un des deux objets combinés par application afin de briser cette symétrie. L'élément évacué pour des fins de bris de symétrie peut être l'argument aussi bien que la fonction, ce qui explique l'indétermination de l'ordre linéaire pour les traits combinés par application fonctionnelle[9].

(66)

a.



b.



En ajoutant les deux opérations d'identification (insertion lexicale), qui permettent d'associer un objet syntaxique à une représentation phonologique ou sémantique via les entrées lexicales, nous obtenons au final les cinq opérations d'interprétation en (67) :

(67) Règles d'interprétation

a. Identification fonctionnelle: S → L ;

où S est une structure syntaxique et L une représentation sémantique.

b. Identification segmentale: S → P ;

[9] Indétermination au niveau translinguistique, bien sûr. Je laisse de côté pour l'instant la façon de déterminer pour une configuration donnée lequel des deux éléments doit être évacué. La question sera (partiellement) résolue plus loin.

où S est une structure syntaxique et P une chaîne de segments

soient A et B des structures syntaxiques ayant reçu une interprétation à l'interface concernée:

c. Application fonctionnelle: {A,B} → APP(A,B) (= A(B) ou B(A))

d. Composition fonctionnelle: {A,{B}} → COMP(A,B)

e. Concaténation: {A,{B}} → PREC(A,B)

Notez le caractère récursif de l'ensemble de règles en (67) : (67c-e) s'appliquent seulement à des objets ayant déjà été interprétés (soit par (67c-e) elles-mêmes, soit par (67a-b)). Il s'ensuit que la première interprétation au bas de l'arbre est toujours obtenue par identification (67a-b). Étant donné ces opérations, je vais maintenant remplacer le principe de Lexicalisation Exhaustive par (68).

(68) Interprétation exhaustive

a. La structure syntaxique de la phrase doit recevoir une interprétation sémantique (via 67a,c,d)

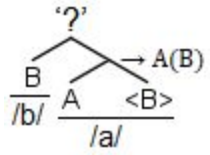
b. La structure syntaxique de la phrase doit recevoir une interprétation phonologique (via 67b,e)

4.1 Le problème de l'enfouissement sémantique

Il reste maintenant un problème théorique à régler concernant les mouvements d'évacuation.

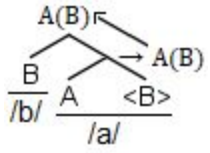
J'ai mentionné au chapitre 2 que l'objet syntaxique résultant d'un mouvement d'évacuation devait d'une façon ou d'une autre « conserver » le contenu sémantique de son nœud fille. Or, on peut voir en (69) qu'aucune des règles de (67) ne permet d'attribuer un contenu sémantique au constituant résultant d'un mouvement d'évacuation.

(69)



Après le mouvement, le contenu sémantique obtenu par application fonctionnelle entre A et B, qui est nécessaire aux computations ultérieures, se trouve « enfoui » à l'intérieur du nouvel objet syntaxique. Une solution potentiellement simple à l'enfouissement sémantique serait de simplement prévoir un mécanisme de projection après le mouvement.

(70)



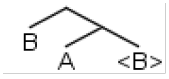
Je vais opter pour une autre solution qui ne nécessite pas de postuler de mécanisme supplémentaire. Pour ce faire, je vais plutôt tirer parti d'une possibilité offerte par la définition de Merge que nous avons adoptée au chapitre 1, soit celle d'une opération prenant deux objets et renvoyant un ensemble non ordonné avec ces deux objets pour éléments :

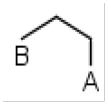
(71) Merge(A,B) = {A,B}

Cette définition de Merge va permettre d'exploiter le jeu des entrées lexicales de façon à obtenir les interprétations phonologiques et sémantiques dans les bonnes configurations. Je rappelle que le mouvement d'évacuation ne laisse pas de trace ou de copie. En ce qui concerne les interfaces, la structure post-mouvement en (72a) est donc équivalente à (72b).

(72)

a. b.





Or, il y a une autre façon autre que le mouvement d'arriver à la structure en (72b). Cette solution découle en fait directement de la définition de Merge en (71).

(73) Self-Merge

a. Merge(A,A) = {A,A} = {A}

b. {A} ≠ A

Les faits en (73) découlent directement de la définition de Merge et des axiomes de la théorie des ensembles. Self-Merge n'est donc pas une opération additionnelle mais une instance

particulière de Merge où A=B.

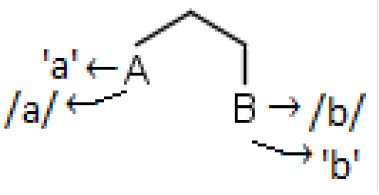
Étant donné que Merge est une opération libre, on peut forcer la syntaxe à effectuer les bonnes applications de Self-Merge, External Merge et Internal Merge par le jeu entre les entrées lexicales (sémantiques et morphologiques) et le principe d'Interprétation Exhaustive. Prenons le fragment de lexique en (74).

(74)

Entrées phonologiques	Entrées sémantiques
<A, /a/>	<{A}, 'a'>
<B, /b/>	<B, 'b'>

Disons que 'a' et 'b' doivent se combiner par application fonctionnelle. Il faut commencer par appliquer Self-Merge à l'un des deux éléments syntaxiques (A ou B) pour que la structure puisse être linéarisée. Si on Self-Merge B (et donc pas A), {B} ne peut pas être interprété par identification fonctionnelle parce que l'entrée sémantique est spécifiée pour B et non {B}.

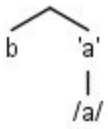
(75)



Les deux fonctions se trouvent donc dans une configuration asymétrique et la structure est

interprétée comme l'instruction de combiner 'a' et 'b' par composition et non par application. Si on Self-Merge plutôt A, {A} est interprété par identification fonctionnelle alors que A est interprété par identification segmentale[10]. Du côté de la sémantique, 'b' et 'a' sont dans une configuration symétrique, donc sont combinés par application fonctionnelle. Du côté phonologique, en revanche, les morphèmes sont dans une configuration asymétrique, ce qui permet d'interpréter l'arbre par concaténation.

(76) a. b.



La conclusion est que quand deux éléments doivent se combiner par application fonctionnelle, l'élément qui précède linéairement est interprété sémantiquement et phonologiquement sur le même niveau d'enchâssement, alors que l'élément qui suit linéairement est interprété sémantiquement à 1 niveau d'enchâssement plus haut que phonologiquement, tel qu'en (77).

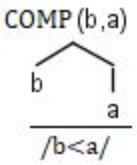
[10] Pour alléger les certaines des représentations à venir, les structures seront parfois abrégées de la façon suivante : les dénotations sémantiques et les formes phonologiques obtenues par identification seront données directement sur le nœud où elles sont obtenues, et l'absence d'indication signifie par défaut que le même nœud syntaxique est interprété sémantiquement et phonologiquement via identification. Par exemple, (76b) est la forme abrégée de (76a).

(77)

Notez que rien ne « force » directement les entrées lexicales pour A et B à être réparties comme en (77). C'est plutôt la pression exercée par l'Interprétation Exhaustive qui oblige le lexique d'une langue à prendre une forme permettant de construire des structures interprétables. Une façon de le voir est que les lexiques sont libres de varier aléatoirement dans leur forme et que les langues naturelles sont un sous-ensemble de tous les lexiques possibles, ceux qui permettent de construire des structures interprétables.

Supposons maintenant que 'a' et 'b' doivent plutôt se combiner par composition fonctionnelle. Chaque trait doit alors être prononcé et interprété sémantiquement sur un même niveau. Le trait correspondant à la fonction input doit ensuite être Self-Merged avant de se combiner à l'autre trait.

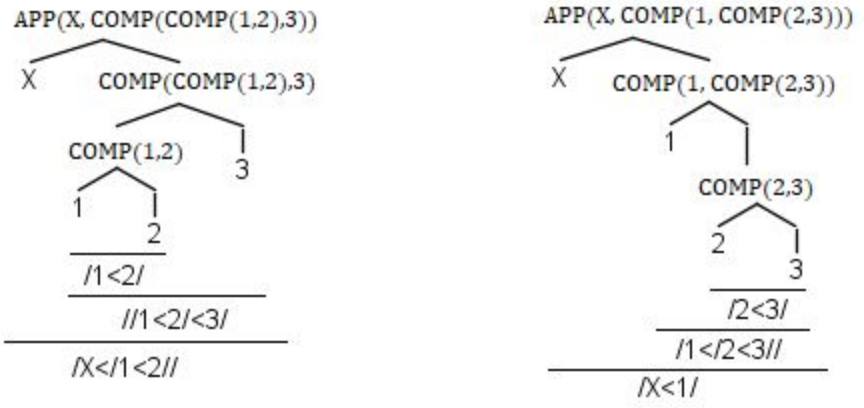
(78)



4.2 Dériver les suffixes décroissants

Je vais maintenant montrer comment le système permet de dériver des suffixes décroissants (X-1-2-3). La représentation (79) montre les deux structures possibles pour l'ordre N-Dem-Num-Adj (X-1-2-3), qui était précédemment indérivable.

(79) a. b.

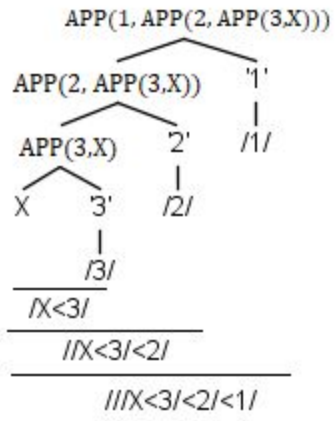


En (79a), 2 et 1 sont d'abord assemblés par composition, après quoi la fonction résultante est elle-même assemblée à 3 par composition. L'ordre des opérations de composition est inversé en (79b). Dans les deux cas, X est seulement introduit à la fin et se combine par application avec la fonction complexe.

Du côté de la phonologie, notez que l'ensemble de règles en (67) permet à la concaténation de

s'appliquer récursivement (e.g. //1<2/<3/ en (79a)) ou non (e.g. /X<1/ en (79b)) dépendamment de quels nœuds se trouvent en relation de c-commande asymétrique. La différence entre l'application récursive et non récursive de la concaténation peut être vue comme faisant des prédictions au niveau de la prosodie, qu'il sera intéressant d'explorer dans le futur. Par exemple, si la concaténation récursive est interprétée comme correspondant à une structure prosodique plus étagée, on pourrait peut-être expliquer le fait que les suffixes semblent soumis au Principe Miroir, contrairement aux mots libres, par le fait que, comme on peut voir en (80), l'ordre Miroir (X-3-2-1) nécessite trois instances consécutives de concaténation récursive, contrairement à l'ordre X-1-2-3 en (79).

(80)



Il reste encore beaucoup d'autres points à couvrir concernant la présente approche, notamment comment reformuler le Superset Principle d'une façon qui soit cohérente avec la présente architecture, ou encore la façon de forcer par les entrées lexicales l'ordre dans lequel doivent

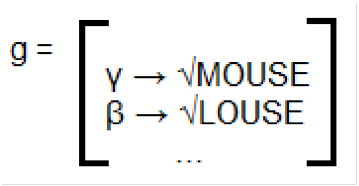
se combiner les traits dans une langue donnée. Je laisse ces questions pour un prochain travail, la conclusion partielle de ce chapitre étant qu'il semble possible de contraindre indirectement la production de Merge par l'interaction entre des contraintes d'interfaces pour dériver les mêmes faits que la Fseq, et que les dispositions prises pour y arriver permettent de résoudre le problème des suffixes décroissants.

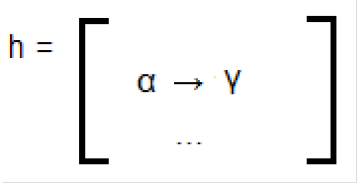
4.3 Infixation et ablaut

Nous avons maintenant les outils nécessaires pour analyser l'infixation et le ablaut, à un détail près : j'ai jusqu'à maintenant supposé que la dénotation d'une racine nominale devait nécessairement être interprétée directement par une entrée lexicale correspondant à une structure idiomatique (e.g. {CHAIS, FEM}-> 'chaise'). Afin de donner au ablaut et à l'infixation un traitement parallèle à celui des suffixes décroissants, il faudra supposer que les éléments syntaxiques plus petits que la racine peuvent également se combiner par application et par composition. Pour ce faire, il suffit de postuler un certain type sémantique, que j'appellerai *r* (pour « racine »). Les éléments de type r sont simplement des variables ou « *placeholders* » entrant dans la dénotation d'items de classe ouverte. Le type r permet par extension l'existence de fonctions de types <r,r> et <r,<e,t>>. Du point de vue du calcul sémantique, ces fonctions se comportent comme n'importe quelle autre, mais leur caractère idiomatique se traduit par le fait qu'elles doivent être définies par extension plutôt que par intension, c'est-à-dire que les outputs de la fonction ne sont pas prédictibles à partir de l'input

et doivent être listés. Pour la dérivation de *mice*, nous aurons donc les fonctions suivantes.

(81) a. b.





Ces fonctions constituent des représentations sémantiques pouvant être obtenues par insertion lexicale comme n'importe quelle autre. La différence est qu'elles correspondent à des éléments plus petits que la racine et que ses outputs sont donc idiosyncratiques. La fonction g est de type <r,<N>>. Elle renvoie une certaine racine nominale en fonction de la variable de type *r* qui lui est donnée en input. Lorsqu'elle reçoit la variable γ, la fonction g renvoie MOUSE; lorsqu'elle reçoit β, elle renvoie LOUSE, et ainsi de suite. La fonction h est de type <r,r>. Elle prend un objet de type r et renvoie une autre variable, qui pourra ensuite être prise comme argument par une autre fonction <r,r> ou encore par une fonction <r,N> telle que g.[11] Voici maintenant les entrées lexicales impliquées dans la dérivation de *mice*.

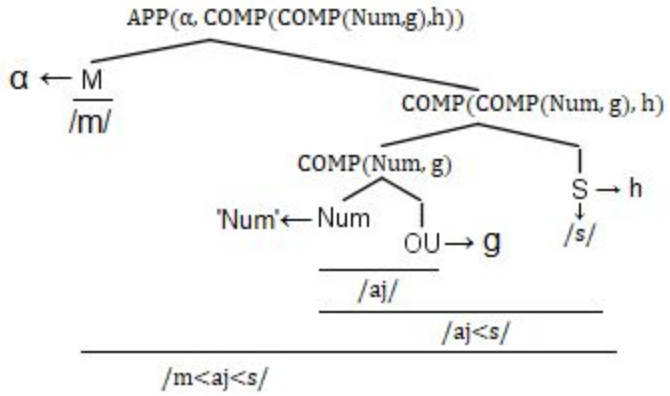
(82)

<M, 'α'>	<M, /m/>
<NUM, 'Num'>	<{NUM,{OU}}, /aj/>
<OU, 'g'>	<OU, /aw/>
<S,'h'>	<S, /s/>

[11] Notez qu'un tel lexique ne contient pas plus (et possiblement moins) d'information qu'un lexique de format plus classique où les paires formes-sens sont données sous forme de liste. En ce qui concerne le geste de postuler un type sémantique en apparence sans contribution sémantique, il devra faire l'objection d'une discussion approfondie que je ne serai pas en mesure de faire ici, mais à première vue je crois qu'il est au pire sans conséquence, au mieux désirable pour des raisons indépendantes.

La dérivation du ablaut en (83) peut donc se faire de façon analogue à celle des suffixes décroissants en (79).

(83)



Pour arriver au sens de ‘mouse.pl’, il faut que la fonction h prenne α comme argument pour renvoyer γ, qui sert à son tour d’input à la fonction g pour renvoyer la racine MOUSE. En (83), les trois fonctions impliquées (h, g et Num) sont combinées par composition pour donner une fonction complexe, et lorsque celle-ci peut finalement prendre α comme argument au haut de l’arbre, cette dernière opération équivaut à appliquer récursivement les fonction h, g et Num, dans cet ordre. La dénotation obtenue au sommet de l’arbre équivaut donc à Num(g(h(α))), c’est-à-dire ‘Mouse.pl’. Du côté de la phonologie, le fait que OU et Num forment un constituant permet de rendre compte du fait que c’est seulement la chaîne de segment /aj/ qui est modifiée en présence du pluriel, et non toute la forme du nom (/mouse/).

En somme, l'infixation est traitée ici comme une instance de suffixes décroissants où l'une des

fonctions impliquées est de type <r,<N>>, et l'ablaut une instance d'infixation où l'infixe a une réalisation supplétive en présence d'un certain trait fonctionnel.

Conclusion

En cherchant à radicalement tirer les conséquences de la SMT en ce qui concerne la nature des traits syntaxiques, j'ai proposé de revoir la relation entre la syntaxe et la composante sémantique. L'architecture proposée réduit les deux interfaces à un ensemble restreint de règles d'interprétation que la syntaxe est chargée d'alimenter. L'interaction entre ces contraintes permet de contenir indirectement la surgénération massive de Merge libre et rendre compte de certaines restrictions distributionnelles classiquement dérivées à l'aide de la Séquence Fonctionnelle. La division explicite entre le contenu sémantique et syntaxique des traits permet de simplifier l'interface à S-M et dispense possiblement de la nécessité d'une composante morphologique.

Bibliographie

Abels, Klaus et Ad Neeleman 2009. Universal 20 without the LCA. J.M. Brucart, A. Gavarro et J. Solà (dir. publ.), *Merging Features – Computation, Interpretation and Acquisition*.
New York : Oxford University Press, pp. 60-79.

Baker, Mark. 1988. I*ncorporation. A Theory of Grammatical Function Changing*. Chicago : University of Chicago Press.

Baunaz, Lena et Eric Lander. 2018. Nanosyntax: The Basics. L. Baunaz, K. De Clercq, L. Haegeman et E. Lander (dir. publ.), *Exploring Nanosyntax.* New York : Oxford University Press, pp. 3-56.

Berwick, Robert et Noam Chomsky. 2011. The Biolinguistic Program: The Current State of its Development. A.-M. Di Sciullo et C. Boeckx (dir. publ.), *The Biolinguistic Enterprise*. Oxford : Oxford University Press, pp.19-41

Bobaljik, Jonathan. 2012. *Universals in Comparative Morphology*. Cambridge, Mass. : MIT Press.

Boeckx, Cedric. 2010. A Tale of Two Minimalisms. Reflections on the plausibility of crash-proof syntax, and its free-merge alternative. M. Putnam (dir. publ.), *Exploring Crash-proof Grammars.* Amsterdam/Philadelphie : John Benjamins, 105-123.

Boeckx, Cedric. 2015. *Elementary Syntactic Structures. Prospects of A Feature-Free Syntax*. Cambridge : Cambridge University Press.

Bye, Patrik et Peter Svenonius. 2010. Exponence, phonology and non-concatenative

morphology. Manuscrit, Université de Tromsø/CASTL.

Caha, Pavel. 2013. Explaining the structure of case paradigms by the mechanism of nanosyntax. *Natural Language & Linguistic Theory* 31, pp. 1015-1066

Chomsky, Noam. 2000. Minimalist inquiries: the framework. R. Martin, D. Michaels et J. Uriagereka (dir. publ.), *Step by Step. Essays on Minimalist Syntax in Honor of Howard Lasnik*. Cambridge, Mass.: MIT Press, pp. 89-155.

Chomsky, Noam. 2001. Derivation by Phase. M. Kenstowicz (dir. publ.), *Ken Hale: A Life in Language*. Cambridge, Mass.: MIT Press, pp. 1-52.

Chomsky, Noam. 2013. Problems of Projection. *Lingua* 130, pp. 33-49.

Chomsky, Noam. 2015. Problems of Projections: Extensions. E. di Domenico, C. Hamann et S. Matteini (dir. publ.), *Structures, Strategies and Beyond: Studies in Honour of Adriana Belletti*. Amsterdam/Philadelphie : John Benjamins, pp. 1-16.

Cinque, Guglielmo. 1999. *Adverbs and functional heads*. New York : Oxford University Press.

Cinque, Guglielmo. 2005. Deriving Greenberg's Universal 20 and Its Exceptions. *Linguistic Inquiry* 36: 3, pp. 315–332.

Collins, Chris. 2017. Merge (X,Y) = {X,Y}. L. Bauke, A. Blümel et E. Groad (dir. publ.), *Labels and Roots*. Berlin : Mouton de Gruyter, pp. 47-68.

De Clercq, Karen et Guido Vanden Wyngaerd. 2017. Comparative Root Suppletion: DM vs Nanosyntax. TIN-dag, Utrecht.

Ernst, Thomas. 2002. *The Syntax of Adjuncts*. Cambridge : Cambridge University Press.

Fábregas, Antonio. 2007. The Exhaustive Lexicalisation Principle. *Nordlyd* 34:2, pp. 165-199.

Fábregas, Antonio et Sergio Scalise. 2012. *Morphology: From Data to Theories*. Édimbourg : Edimburgh University Press.

Greenberg, Joseph H. 1963. Some Universals of Grammar with Particular Reference to the Order of Meaningful Elements. J. H. Greenberg (dir. publ.), *Universals of Human Language*. Cambridge, Mass. : MIT Press, pp. 73-113.

Harley, Heidi et Rolf Noyer. 1999. State-of-the-Article: Distributed Morphology. *Glot International* 4:4, pp. 3-9.

Heim, Irene et Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Oxford : Blackwell.

Kayne, Richard. 1994. *The Antisymmetry of Syntax*. Cambridge, Mass. : MIT Press.

Kayne, Richard. 2016. What is Suppletive Allomorphy? On *went* and on **goed* in English. Manuscrit, New York University.

Nilsen, Øystein. 2003. *Eliminating Positions: Syntax and Semantics of Sentence Modification*. Thèse de doctorat, Université d'Utrecht.

Pantcheva, Marina. 2011. *Decomposing Path. The Nanosyntax of Directional Expressions*. Thèse de doctorat, Université de Tromsø.

Ramchand, Gillian et Peter Svenonius. 2014. Deriving the functional hierarchy. *Language Sciences* 46, pp. 152–174.

Rizzi, Luigi. 1997. The fine structure of the left periphery. L. Haegeman (dir. publ.), *Elements of Grammar*. Dordrecht: Kluwer Academic Publishers, pp. 281-337.

Starke, Michal. 2001. *Move dissolves into Merge*. Thèse de doctorat, Université de Genève.

Starke, Michal. 2009. Nanosyntax: A Short Primer t a New Approach to Language. *Norldyd: Special Issue on Nanosyntax* 36, pp. 1-6.

Starke, Michal. 2011a. Towards an elegant solution to language variation: Variation reduces to the size of lexically stored trees. Lingbuzz/001183

Starke, Michal. 2011b. Nanosyntax, part I. Lecture series at GIST, Ghent.

Starke, Michal. 2018. Complex Left Branches, Spellout, and Prefixes. L. Baunaz, K. De Clercq, L. Haegeman et E. Lander (dir. publ.), *Exploring Nanosyntax*. New York : Oxford University Press, pp. 239-249.

Steedman, Mark. 1996. A very short introduction to CCG. Manuscrit, University of Edimburgh.

Vanden Wyngaerd, Guido. 2018. The feature structure of pronouns: a probe into multidimensional paradigms. L. Baunaz, K. De Clercq, L. Haegeman et E. Lander (dir.

publ.), *Exploring Nanosyntax*. New York : Oxford University Press, pp. 277-304.